Survey on the Efficacy of Agile Development – Zachary Zerafa (24557656)

Abstract

In a world where technological solutions prevail, the demand for software development and application products is increasing exponentially. To satisfy this, software companies have developed and employed various strategies of systematic, corporate processes to ensure that the product deadlines are reached and that the client's software requirements are fufilled. This gave rise to agile development; a software development ideology best suited for small teams that follows an iterative structure of repeatedly updating backlogs and gathering with a team to collaborate on a project.

This system was formalised in 2001, and since then the nature and magnitude of programming has evolved immensely. In this survey various agile methods will be evaluated and a conclusion on the best apporach to modern agile development will be reached.

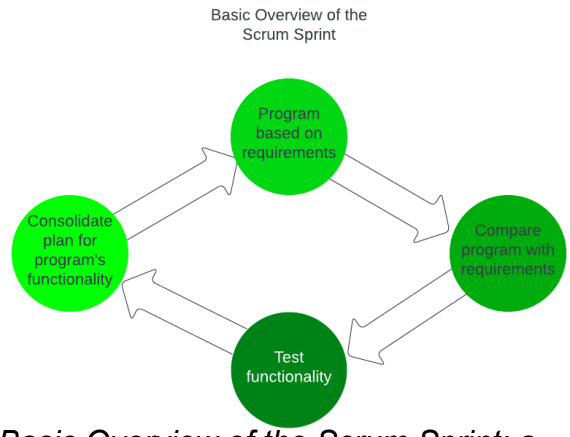
1 – Introduction

What is Agile Development?

Agile development was a software development philosophy formalised in 2001 to retort to the unprecedented software requirements that clients were requesting. It is often described as iterative, meaning that it works based on repeating a series of steps frequently in different stages of the program's development. Agile development is included in many methodologies, such as the scrum and kanban methods.

Practical use of agile development

These attributes aim to equip software developing teams with a process that is extremely efficient when confronted with changes of requirements, as well as accurately meeting said requirements. Following the Dot-Com bubble (Al-Zewairi et al., 2017), these qualities became sought by businesses that aimed to establish an online presence and further contributed to agile development's use in mainstream software developing firms.



Basic Overview of the Scrum Sprint; a visual demonstration of a scrum sprint, demonstrating the iterative nature of a sprint

2 – Scrum

Introduction to Scrum

Scrum is a method is designed so that a stakeholder is able to point out undesirable effects of prototypes early and programmers can mitigate the issues in a timely fashion. This is due to the use of frequent conferences as an opportunity to reflect on how well a program's functionality

will suit the needs of a stakeholder. This process divided into “sprints” (Kumar et al. 2012), which are time frames in which small groups of developers (of ten from “five to nine people”, but may be well over (Kumar et al. 2012)) are to work on their project at hand and at the end of each sprint, a prototype is demonstrated with the stakeholders, who will judge the precision to which the demonstration suits the user requirements. A scrum-master is tasked with managing the scrum session and guiding developers in following the user requirements discussed in scrum meetings, weaving the entire methodology together with the developing team.

The Benefits

Due to the consistency and frequency of these meetings, it is a useful technique in maintaining strong and timely communication with the stakeholders, who define the scope of the software. Evidently, an analysis by A .Ahmed shows that 50% (Ahmed et al., 2010, Kumar et al. 2012) of scrum activities (including scrum) involve stakeholders. Additionally, company PulpCo (Paasivaara et al., 2009) compared their results before and after employing iterative sprints. It was noted that before sprints, programmers often worked individually and drifted away from requirements more frequently, whilst by iteratively going through the cycle of sprints they kept strong contact with stakeholders and minimised the amount of rework due to the consistency of communication. The scrum process was proven to bind the team closer to the requirements and increase workplace efficiency.

The Tradeoff

The scrum method would require considerable effort from stakeholders to ensure that requirements are satisfied and that the requirements are within the range of development teams to complete. It also places high pressure on the scrum master to ensure that the scrum process is proceeding as intended, without sufficient effort, the methodology becomes inefficient. However by keeping stakeholders networked and having a strong scrum-master, their requirements are better fufilled.

3-Kanban

Introduction to Kanban

Keeping track of workflow in programming is essential; overlapping tasks is a hazard that disrupts efficiency and adds surplus cost. However it is easily avoidable in agile environments with iterative techniques to keep communication in check. The Kanban method originates from 1940s Toyota employee Taiichi Ohno (Kirovska et al., 2015) in a, the same method can be applied by the creation of user stories (structured user requirements with reasoning for their implementation), allocation of these stories to development teams, ranking of importance for each user story, and further atomisation of these user stories, and finally sorting out which user stories are fufilled, in progress, or yet to program (Al-Zewairi et al., 2017).

Effects

The purpose of this is to narrow the focus for each group of developers on their respective tasks, quantify the real-time workflow (Alaidaros et al., 2021)of all areas of the application and

shift resources appropriately. It is a form of iterative communication applied in a way unique from scrum; replacing constant intervals of stakeholder communication with defining the requirements well at the beginning and consistently marking off each requirement fufilled. Though there is less interaction with stakeholders, surprisingly the Kanban method performs better than the scrum method, with a slightly more consistent rate of success (Alaidaros et al., 2021).

Overscoping

Kanban is valued for its ability to quantify requirements and allow for greater resource flexibility with minimal communication with stakeholders, however can be affected by overscoping, which refers to undertaking an “unrealistically grand amount of functions” (Bjarnason et al., 2012) that are beyond the means of the developing team, in either a sense of finance, magnitude, or timing. Kanban’s lack of communication can prove an issue when a team progressively recognises that the user stories may have a scope that is unrealistic to implement under certain conditions, whereas scrum’s more iterative approach allows for these issues to be identified and mitigated earlier.

4 - Extreme Programming

Extreme Programming Process

The agile method also comes in variants that hone in more on software development (Tripp et al., 2016), extreme programming (XP) being the most popular of them all. XP is defined by a strong planning stage, where test functions are delevoped on user stories prior to case-specifc code to analyse the project’s feasibility. Then the code is “refactored” into a form that is more efficient (such as reducing code’s Big O complexity) (What is Extreme Programming (XP)? 2021).From here, developers start “pair-programming” (Tripp et al., 2016); working on their sector of programming in pairs of two, which makes the coding process hastier and more meticulous.

Efficacy

XP allows for the code to be reviewed with ease in comparison to other methods, mostly thanks to pair programming as a method to gain some level of instantaneous feedback. It also offers more flexibility as programming teams are given more independence to work on test cases, reducing the cost and time of backtracking and changing previous code. The case study of Wood and Kleb (Wood & Kleb, 2003, Layman et al 2004) supports this claim, where they analysed a team at NASA use XP with a new language to solve a mathematical problem, concluding that the experiment had double the average productivity. Though the results are succesful, more data may be necessary to back this claim for larger teams.

However the high focus on code quality also leaves possibility for gaps in the software’s design; functionality may be optimum, but if the product is not designed in an easy-to-use manner, stakeholders may be disgruntled.

5 - Lessons Learned and Future Promising Direction

Positive Outcomes

Agile method comes in many different forms but ultimately all benefit from the philosophy of cycling through bursts of iterative processes and working in a lightweight way that allows for adaptability to change. Evidently most types of agile produce desirable results as seen in previous case studies, as the use of placing programmers in collaborative groups (such as in scrum development teams and pair-programming teams for XP) boosts efficiency and helps to rectify concerns autonomously before even involving stakeholders.

Negative Outcomes

The adverse of scrum's heavy involvement with stakeholders implies that their absense and the absense of scrum-master creates confusion and extra cost. Kanban's independent scheduling techniques may reduce the need of tight stakeholder communication, but it delays the process of mitigating overscopes, which creates inefficiency. And though XP helps guide code optimisation, its lack of focus on design can lead to a project that meets fewer requirements. The recurring pattern is that the core concept behind each method is geared for agility, but in doing so leaves open room for possible surplus cost if these risks are not acknowledged.

Optimised Solution for Future

Ideally a modern software company would want to apply a concoction of agile development based methods in unison to help reduce the risk factors of religiously following a single method. There are hybrid agile development methods such as scrumban (a method that balances a strong network with stakeholders while also valuing the idea of kanban's independent, dynamic backlogging process) that are modern solutions businesses employ Alaidaros et al., 2021).

Reference

Al-Zewairi, M., Biltawi, M., Etaiwi, W., & Shaout, A. (2017). Agile Software Development Methodologies: Survey of Surveys. *Journal of Computer and Communications*, *05*(05), 74–97. https://doi.org/10.4236/jcc.2017.55007

Bjarnason, E., Wnuk, K., & Regnell, B. (2011). Requirements are slipping through the gaps: a case study on Causes & effects of communication gaps in large-scale software development. *2011 IEEE 19th International Requirements Engineering Conference*. https://doi.org/10.1109/re.2011.6051639

Bjarnason, E., Wnuk, K., & Regnell, B. (2012). Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering. *Information and Software Technology*, *54*(10), 1107–1124. https://doi.org/10.1016/j.infsof.2012.04.006

Kumar, G., & Bhatia, P. K. (2012). Impact of Agile Methodology on Software Development Process, 1–16.

Layman, L., Williams, L., & Cunningham, L. (2004). Exploring extreme programming in context: An industrial case study. *Agile Development Conference*, 1–10. https://doi.org/10.1109/adevc.2004.15

Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2009). Using Scrum in distributed agile development: A multiple case study. *2009 Fourth IEEE International Conference on Global Software Engineering*, 1–10. https://doi.org/10.1109/icgse.2009.27

Tripp, J., Rienemschneider, C., & Thatcher, J. (2016). Job satisfaction in Agile Development Teams: Agile Development as work redesign. *Journal of the Association for Information Systems*, *17*(4), 267–307. https://doi.org/10.17705/1jais.00426

Wood, W. A., & Kleb, W. L. (2003). Exploring XP for scientific research. *IEEE Software*, *20*(3), 30–36. https://doi.org/10.1109/ms.2003.1196317

Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., & Sarwar, S. Z. (2010). Agile software development: Impact on productivity and Quality. *2010 IEEE International Conference on Management of Innovation & Technology*. https://doi.org/10.1109/icmit.2010.5492703

What is Extreme Programming (XP)? Agile Alliance | Extreme Programming (XP). (2021, March 10). Retrieved March 15, 2023, from https://www.agilealliance.org/glossary/xp/#q=~(infinite~false~filters~(postType~(~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1

Alaidaros, H., Omar, M., & Romli, R. (2021). The state of the art of Agile Kanban Method: Challenges and opportunities. *Independent Journal of Management & Production*, *12*(8), 2535–2550. https://doi.org/10.14807/ijmp.v12i8.1482

Kirovska, N., & Koceski, S. (2015). *Usage of Kanban Methodology at Software Development Teams*, 1–10.